Alexander Garcia May 30, 2017

Assignment-1

1. Let $f(x) = e^x \cos 2x$.

   (a) The definition of the Taylor polynomial is as follows:

$$T_n(x) = \sum_{k=0}^{n} \frac{(x - x_0)^k}{k!} f^{(k)}(x_0)$$

   In this case, $n = 4$, and $x_0 = 0$.
   We start by finding $f^{(1)}(x), f^{(2)}(x), f^{(3)}(x)$, and $f^{(4)}(x)$

   $f^{(1)}(x) = -2e^x sin(2x) + e^x cos(2x) = e^x(cos(2x) - 2sin(2x))$
   $f^{(2)}(x) = e^x(-2sin(2x) - 4cos(2x)) + e^x(cos(2x) - 2sin(2x)) = -e^x(4sin(2x) + 3cos(2x))$
   $f^{(3)}(x) = -e^x(8cos(2x) - 6sin(2x)) - e^x(4sin(2x) + 3cos(2x)) = e^x(2sin(2x) - 11cos(2x))$
   $f^{(4)}(x) = e^x(4cos(2x) + 2sin(22sin(2x)) + e^x(2sin(2x) - 11cos(2x)) = e^x(24sin(2x) - 7cos(2x))$

   Evaluating these functions at the center point, we get:

   $f(0) = e^0 cos(0) = 1$
   $f^{(1)}(0) = e^0(cos(0) - 2sin(0)) = 1$
   $f^{(2)}(0) = -e^0(4sin(0) + 3cos(0)) = -3$
   $f^{(3)}(0) = e^0(2sin(0) - 11cos(0)) = -11$
   $f^{(4)}(0) = -e^0(24sin(0) - 7cos(0) = -7$

   We now have all the pieces we need to determine the polynomial expansion.

$$P_n(x) = \frac{x^0}{0!} f(0) + \frac{x^1}{1!} f^{(1)}(0) + \frac{x^2}{2!} f^{(2)}(0) + \frac{x^3}{3!} f^{(3)}(0) + \frac{x^4}{4!} f^{(4)}(0)$$
$$= 1 + x - \frac{3}{2}x^2 - \frac{11}{6}x^3 - \frac{7}{24}x^4$$

   (b) By definition, the derivitive form of the remainder is

$$R_n(x) = \frac{(x - x_0)^{n+1}}{(n+1)!} f^{(n+1)}(c)$$

   where c is close to x_0 .
   In this case, we have $n = 4, x_0 = 0$ from the previous question.

$R_n(x) = \frac{x^5}{5!}f^{(5)}(c)$
$f^{(5)}(c) = e^x(48cos(2x) + 14sin(2x)) + e^x(24sin(2x) - 7cos(2x))$
$= e^x(41cos(2x) + 38sin(2x))\,]_{\,x=c}$

Therefore,

$$R_4(x) = \frac{x^5}{120}e^c(41cos(2c) + 38sin(2c))$$

(c) We know the remainder, $R_4(x)$ from part (c). Given a closed interval, we can find the maximum error by finding the maximum remainder in this interval. To do this, we must find its derivative, whose zeroes will show all maxima and minima.
$f^{(6)}(x) = e^x(-82sin(2x) + 76cos(2x)) + e^x(41cos(2x) + 38sin(2x)) = e^x(117cos(2x) - 44sin(2x))$

Now find zeroes.
$e^x(117cos(2x) - 44sin(2x)) = 0$
$117cos(2x) - 44sin(2x) = 0$
$\frac{117}{44} = \frac{sin(2x)}{cos(2x)}$
$x = \frac{1}{2}tan^{-1}(\frac{117}{44}) \approx 0.606$

Our candidate points are then $\{-\frac{\pi}{4}, 0.606, \frac{\pi}{4}\}$

$f^{(5)}(\frac{\pi}{4}) = 38 * e^{\frac{\pi}{4}} \approx 83.3$
$f^{(5)}(-\frac{\pi}{4}) = -38 * e^{\frac{\pi}{4}}$
$f^{(5)}(0.606) = e^{0.606}(41cos(2 * 0.606) + 38sin(2 * 0.606)) \approx 91.6$

As $f^{(5)}(0.606)$ is clearly the max, this is what we use for determining the error bound.

$$max(R_4(x)) = \frac{(\frac{\pi}{4})^5}{5!} * 91.6$$

$$\approx 0.228$$

(d) Figure 1 shows the actual function compared to its 4^{th} degree Taylor expansion over the range $x \in [\frac{-\pi}{4}, \frac{\pi}{4}]$. Figure 2 shows the maximum error between the two, which agrees with the upper bound calculated in part (c).

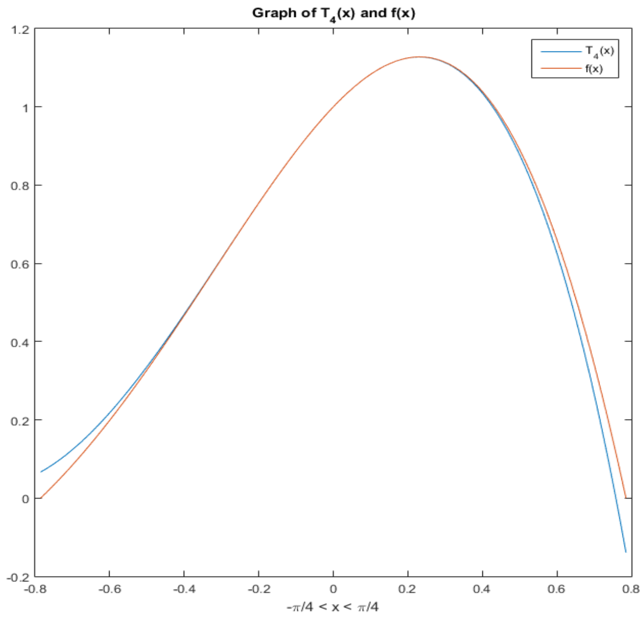

Figure 1: Taylor expansion and initial function

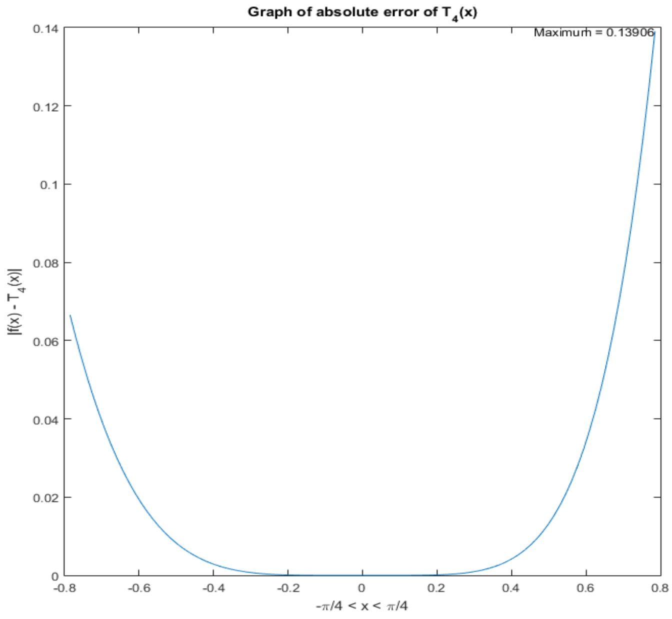

Figure 2: Difference between the Taylor expansion and the initial function

2. $b = 10.\overline{110}_2 \;\; = 2^1 + (0.\overline{110})_2$

   $z = 0.\overline{110}_2$

   $z = 0.110\overline{110})2$

   $2^3 * z = 110.\overline{110}_2 = 2^2 + 2^1 + z$

   $8z = 6 + z$

   $z = \frac{6}{7}$

   $b_{10} = 2 + z_{10} = 2 + \frac{6}{7} = \frac{18}{7} = 2.\overline{857142}$

3. 6.7_{10}

   $6_{10} = 110_2$

   $$\begin{array}{ll} \frac{14}{10} = d_1 + \frac{d_2}{2} + \cdots & d_1 = 1 \\ \frac{4}{5} = d_2 + \frac{d_3}{2} + \cdots & d_2 = 0 \\ \frac{8}{5} = d_3 + \frac{d_4}{2} + \cdots & d_3 = 1 \\ \frac{6}{5} = d_4 + \frac{d_5}{2} + \cdots & d_4 = 1 \\ \frac{2}{5} = d_5 + \frac{d_6}{2} + \cdots & d_5 = 0 \\ \frac{4}{5} = d_6 + \frac{d_7}{2} + \cdots & d_6 = 0 \end{array}$$

   The value generating d_6 is the same as the one generating d_2 , and thus we are in a loop generating the repeating pattern 0110.

   $6.7_{10} = 110.1\overline{0110}_2 = 1.101\overline{0110} * 2^2$

   The IEEE representation of this number designates one signed bit (in this case it is 0, as 6.7¿0), 8 for the exponent (in this case 2), and 23 for the mantissa (in this case $101\overline{0110}$. The 23^{rd} digit of the mantissa is 0, so the number is truncated when stored.

   The exponent that is stored is the real exponent + a bias, which in this case is 127.

   $fl(6.7)$ is stored in the machine as

0	10000001	10101100110011001100110
sign	exponent + bias	mantissa

   Converting the stored number back to decimal gives us $2^{129-127} * q$

   $q = 1 + 2^{-1} + 2^{-3} + 2^{-5} + 2^{-6} + 2^{-9} + 2^{-10} + 2^{-13} + 2^{-14} + 2^{-17} + 2^{-18} + 2^{-21} + 2^{-22} \approx 1.675$

   $fl(6.7) = 2^2 * q \approx (6.7 + 1.90735 * 10^{-7})$

   The relative error $d = \frac{6.7 - fl(6.7)}{6.7} = 2.84679 * 10^{-8}$

   ϵ_{mach} for a 32 bit float is $\frac{1}{2^{23}} \approx 1.19 * 10^{-7}$

   $\frac{1}{2} * \epsilon_{mach} \approx 5.96 * 10^{-8}$

   Crucially,

   $$\frac{1}{2} * \epsilon_{mach} \leq d$$

4. Holmes 1.5

   (a) According to the IEEE double precision standard, 16_{10} is stored as
   $1 * 2^5 =$

0	10000000100	$0 \cdots 0$
sign	exponent + bias of 1023	52 bit mantissa

If we let d_n represent the n^{th} digit in the mantissa, then the last digit stored in the machine is $d_{52} = 0$. By using the round-to-nearest rule, if d_{53} is a 1, we increment d_{52} by one, then chop. This, however, would change the value of the stored number. If d_{53} is 0, however, the number is simply chopped after d_{52} with no modifications. In other words, if $d_1 d_2 \cdots d_{52} d_{53} == 0$, then $d_{54} d_{55} \cdots d_\infty$ can take any value they like. More importantly, the upper bound of this value is when $d_{54} = d_{55} = \cdots = d_\infty = 1$

$$d_{54} d_{55} \cdots d_\infty = \sum_{i=54}^{\infty} 2^{-i} = \frac{1}{9007199254740992} \approx 1.11 * 10^{-}16$$

This makes the upper bound of 16 $R = 16 + \frac{1}{9007199254740992} \approx 16.000000000000000111$

The lower bound of 16 can be done by finding the lowest value at which $15.xxx$ rounds up to 16. In order to round up at all using the round-to-nearest rule, we know that d_{53} of $15.xxx$ must be a 1. This will cause d_{52} to increase by 1, and the number will then be chopped. In order for the entire sequence $d_1 d_2 \cdots d_{52}$ to go to zeros, they must initially all be 1, and the overflow will convert $15.xxx$ into $16.000\cdots$, as well as increase the exponent. This value would therefore be stored as

$$\begin{array}{c|c|c} 0 & 10000000010 & 1\cdots1 \\ \text{sign} & \text{exponent + bias of 1023} & \text{52 bit mantissa} \end{array}$$

In this case, the values of $d_{54} d_{55} \cdots d_\infty$ are arbitrary, and their lower bound would be $d_{54} = d_{55} = \cdots = d_\infty = 0$. So, the lowest value we can take is

$$L = 15 + \sum_{i=1}^{53} 2^{i} = \frac{144115188075855871}{9007199284740992} \approx 15.9999999999999999989$$

(b) The same process can be used to find the upper and lower bounds of 50, but is slightly altered because of the IEEE representation.

The upper bound is found the exact same way.

$$R = 50 + \sum_{i=54}^{\infty} 2^{-i} = \frac{450359962737049601}{9007199254740992} \approx 50.000000000000000111$$

However, the lower bound is slightly altered, due to the binary form of 50.

$50 = 110010 = 1.1001 * 2^5$

$$\begin{array}{c|c|c} 0 & 10000000100 & 10010\cdots0 \\ \text{sign} & \text{exponent + bias of 1023} & \text{52 bit mantissa} \end{array}$$

Here, the lower bound would be $49.xxx$, where all decimal places in the binary form are 1, through the same logic as part(a).

$$\begin{array}{c|c|c} 0 & 10000000100 & 100011\cdots1 \\ \text{sign} & \text{exponent + bias of 1023} & \text{52 bit mantissa} \end{array}$$

Only the values of d_6 through d_{52} can be set to 1, as any other modifications would not represent the number $49.xxx$. Because of this, the lower bound is

$$L = 49 + \sum_{i=1}^{46} 2^{-i} = \frac{3518437208883199}{70368744177664} \approx 49.999999999999985$$

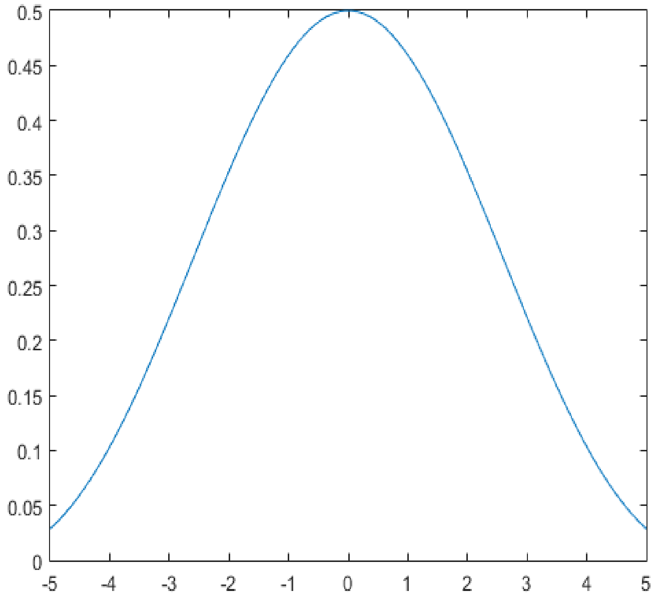

Figure 3: Graph of $\frac{1-cos(x)}{x^2}$

5. It is known that $cos(0) = 1$, so as $x \to 0^+$, the numerator of this function becomes extremely small. When the machine subtracts these two quantities, there is a point at which catastrophic collision begins, which introduces significant error. When the number of ones in the binary form of $cos(x)$ is greater than 52 (the maximum number of digits in an IEEE double precision number), the result will dip below the underflow limit and is seen by the machine as zero.

$$cos(10^{-8}) \approx 0.9999999999999999_{10} \approx 0.111 \cdots 101_2$$

At $x = 10^{-8}$, there are 54 leading ones in the binary form of $cos(x)$, so the machine computes the numerator as zero. The denominator, however, can still be accurately computed as x^2, so the result is simply 0. At slightly larger values of x ($x = 10^{-7}$), there are less than 52 ones, so the machine can still do meaningful subtraction, albeit with minimal significance in the result.

6. Assume single-precision IEEE arithmetic. Assume that the round-to-nearest rule is used with one modification: if there is a tie then the smaller value is picked (this rule for ties is used to make the problem easier).

   (a) For what real numbers x will the computer claim that the inequalities $1 < x < 2$ hold?

   (b) For what real numbers x will the computer claim $x = 4$?

   (c) Suppose it is stated that there is a floating point number x^* that is the exact solution of $x^2 - 2 = 0$. Why is this not possible? Also, suppose x_L^* and x_R^* are the floats to the left and right of $\sqrt{2}$ respectively. What is the value of $x_R^* - x_L^*$?

7. (a) A problem is ill-conditioned if its solution is highly sensitive to small changes in the input data. True or False?
   True. An ill-conditioned problem will grow a slight error in input data to a large error in later stages

   (b) Using higher-precision arithmetic will make an ill-conditioned problem better conditioned. True or False?

(c) If two real numbers are exactly representable as floating-point numbers on a finite-precision machine, then so is their product. True or False?

(d) Consider the sum

$$S = \frac{1}{x+1} + \frac{1}{x-1}, \quad x \neq 1.$$

For what range of values is it difficult to compute S accurately in a finite-precision system? How will you rearrange the terms in S so that the difficulty disappears?

(e) In a finite-precision system with UFL $= 10^{-40}$, which of the following operations will incur an underflow?

i. $\sqrt{a^2+b^2}$, with $a = 1$, $b = 10^{-25}$.

ii. $\sqrt{a^2+b^2}$, with $a = b = 10^{-25}$.

iii. $(a \times b)/(c \times d)$, with $a = 10^{-20}$, $b = 10^{-25}$, $c = 10^{-10}$, $d = 10^{-35}$.